IN THE CLAIMS

1. (Currently Amended) A system for instruction memory storage and processing in a computing device having a processor, the system being based on backwards branch control information, the system comprising:

a dynamic loop buffer (DLB) organized as a direct-mapped structure, said DLB being a tagless array of data;

a DLB controller having a primary memory unit partitioned into a plurality of banks, wherein said controller controls the state of the instruction memory storage system, accepts a program counter address (pc address) as an input, and outputs distinct signals;

an address register located in the memory of said computing device, wherein said address register is a staging register for said program counter address, and an instruction fetch process takes two cycles of said processor; and

a bank select unit for serving as a pc-address decoder to accept the program counter address and to output a bank enable signal for selecting a bank in a primary memory unit, and a decoded address for access within the selected bank, wherein the system is configured to:

fetch instruction packets from the primary memory unit,

write a copy of said instruction packets into the DLB,

set a valid bit to assert validity of said DLB,

enter an ACTIVE state when a backward branch is taken within a range of a loop,

enter an IDLE state when said backward branch completes a loop and a change of flow branch is taken out of range of said loop, and

enter an OVERFLOW state if the DLB fills up and said change of flow branch is not taken; and

if in the IDLE state, enter the ACTIVE state if a last loop captured in the DLB is repeated.

2. (Original) The system of claim 1, further comprising:

a first multiplexor for selecting amongst a plurality of banks of said primary memory unit;

a second multiplexor for selecting between the DLB and the primary memory unit output;

a latch for staging the data; and

a data-out register located in the main memory of said computing device, wherein said data-out register receives instruction data for said processor at the decode stage.

3. (Original) The system of claim 1, wherein said primary memory unit is one of a static random access memory (SRAM) or a dynamic random access memory (DRAM).

4. (Original) The system of claim 1, wherein said DLB is made of energy efficient growable register arrays (GRAs).

5. (Original) The system of claim 1, wherein said DLB is made of energy efficient register array cells.

6. (Original) The system of claim 2, wherein said distinct signals output from said DLB controller include:

signals to enable/disable said processor to access said primary memory unit, to read the DLB and to write the DLB;

decoded address signals for said processor to access said DLB; and

select signals to select a first multiplexor.

7. (Original) The system of claim 1, wherein said DLB further includes a read or write port and accepts three input signals.

8. (Currently Amended) A method of instruction memory storage and processing in a computing device having a processor, a dynamic loop buffer (DLB), and a DLB controller, the method being based on backwards branch control information, the method comprising the steps of:

measuring cycles from a valid program counter address latched from an address register to valid data;

latching said program counter address said DLB controller and a bank select unit from said address register at a rising edge of every clock cycle;

said DLB controller and said bank select unit processing signals for data access during a first half of the clock cycle;

latching signals at the DLB and at a primary memory unit on the falling edge of the clock cycle; ~~and~~

completing and output data in a data register in one clock cycle;

fetching instruction packets from the primary memory unit,

writing a copy of said instruction packets into the DLB,

setting a valid bit to assert validity of said DLB,

entering an ACTIVE state when a backward branch is taken within a range of a loop,

entering an IDLE state when said backward branch completes a loop and a change of flow branch is taken out of range of said loop;

entering an OVERFLOW state if the DLB fills up and said change of flow branch is not taken; and

if in the IDLE state, entering the ACTIVE state if a last loop captured in the DLB is repeated.

9. (Original) A method of claim 8, further comprising a step of latching data from a data register to a data-out and to the DLB for storing at the falling edge of the clock, when said controller is in a FILL state.

10. (Currently Amended) An instruction buffer managing system in a computing system including a processor, said system comprising:

an instruction address input for receiving one or more program counter values from said processor;

a controllable path between the instruction address input and an instruction address output that passes program counter values to an instruction memory, the instruction memory being connected to the instruction address output, wherein said program counter values received at the instruction address input are monitored;

a controller for defining a range of instructions, the range of instructions comprising a loop; and

a filler for filling a buffer with each instruction of a range in a first pass, except for at least one unfilled instruction, wherein the program counter values on one or more subsequent loop iterations are continuously monitored until an unfilled instruction is encountered and a buffer manager provides said unfilled instruction, wherein the system is configured to:

fetch instruction packets from a primary memory unit,

write a copy of said instruction packets into the buffer,

set a valid bit to assert validity of said buffer,

enter an ACTIVE state when a backward branch is taken within a range of a loop,

enter an IDLE state when said backward branch completes a loop and a change of flow branch is taken out of range of said loop, and

enter an OVERFLOW state if the buffer fills up and said change of flow branch is not taken; and

if in the IDLE state, enter the ACTIVE state if a last loop captured in the buffer is repeated.

11. (Original) A system of claim 10, wherein said unfilled instructions are within a sub range defined by a forward non-consecutive control flow instruction.

12. (Original) A system of claim 10, wherein said unfilled instructions are within a sub range defined by a backward non-consecutive control flow instruction.

13. (Original) A system of claim 10, wherein an instruction address of said unfilled instructions fall outside a sequential address associated with the buffer and are therefore omitted.

14. (Original) A system of claim 13, wherein the buffer manager takes unfilled instruction from the instruction memory.

15. (Original) A system of claim 10, wherein instructions in the buffer are retained after the processor fetches instructions outside the range.

16. (Currently Amended) A method of changing states of a dynamic loop buffer (DLB) controller having four states, to enable limiting a use of energy in a computing device having a processor, a DLB, and the DLB controller, while said computing device is performing instruction

memory storage and processing using backwards branch control information, the method comprising the steps of:

in an initial IDLE state, determining if a back branch was taken;

entering a FILL state if the determination is positive, and

if the determination is negative, determining if an unfilled entry was hit;

entering the FILL state if the determination is positive, and

if the determination is negative, determining if a filled entry was hit; and

entering an ACTIVE state if the determination is positive, ~~and~~

if the determination is negative, returning to the IDLE state; and

if in the IDLE state, enter the ACTIVE state if a last loop captured in the DLB is repeated.

17. (Original) The method of claim 16, further comprising the steps of:

in the FILL state, determining if an address is out of range;

entering the IDLE state if the determination is positive, and if the determination is negative, determining if a filled entry was hit,

entering the ACTIVE state if the determination is positive, if the determination is negative, determining if an offset is greater than a physical address; and

entering an OVERFLOW state if the determination is positive,

if the determination is negative returning to the FILL state.

18. (Original) The method of claim 17, further comprising the steps of:

In the ACTIVE state determining if the address is out of range;

entering the IDLE state if the determination is positive, and if the determination is negative, determining if an unfilled entry was hit; and

entering the FILL state if the determination is positive, and if the determination is negative, determining if an offset is greater than a physical address; and

entering the OVERFLOW state if the determination is positive, and

if the determination is negative returning to the ACTIVE state.

19. (Original) The method of claim 17, further comprising the steps of:

in the OVERFLOW state, determining if an unfilled entry was hit;

entering an FILL state if the determination is positive, and if the determination is negative, determining if a filled entry was hit; and

entering the ACTIVE state if the determination is positive, and if the determination is negative, returning to the OVERFLOW state.

20. (Original) A method of changing states of a dynamic loop buffer (DLB) controller having four states, to enable limiting a use of energy in a computing device having a processor, a DLB, and the DLB controller, while said computing device is performing instruction memory storage and processing using backwards branch control information, the method comprising the steps of:

in a FILL state

fetching instruction packets from a primary memory unit,

writing a copy of said instruction packets into a DLB,

setting a valid bit to assert validity of said DLB,

entering an ACTIVE state when a backward branch is taken within a range of a loop,

entering an IDLE state when said backward branch completes a loop and a change of flow branch is taken out of range of said loop, and

entering an OVERFLOW state if the DLB fills up and said change of flow branch is not taken;

in the ACTIVE state

presenting instruction fetches to the DLB,

entering the IDLE state if said backward branch is not taken and a change of flow branch is taken out of range,

entering the FILL state if the buffer is not full, if said backward branch is within the loop,

entering an OVERFLOW state if the buffer is full,

entering the FILL state if the buffer is full, and

entering the OVERFLOW state if the end of the buffer is reached said and change of flow branch is not taken;

in the OVERFLOW state

reading instruction data packets from a primary memory unit of said computing device,

entering the IDLE state after performing a jump or a branch instruction out of range of said loop;

entering the ACTIVE state if said backward branch is taken,

entering the ACTIVE state if a backward branch within the range of said loop is taken and data is valid, and

entering the FILL state if a backward branch within the range of said loop is taken and data is invalid; and

in the IDLE state
accessing instruction data packets from the primary memory unit,
entering the FILL state if a new backward branch is detected, and

entering the ACTIVE state if a last loop captured in the dynamic loop buffer is repeated.

21. (Original) A computer program device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps for a method of changing states of a dynamic loop buffer (DLB) controller having four states, to enable limiting a use of energy in a computing device having a processor, a DLB, and the DLB controller, while said computing device is performing instruction memory storage and processing using backwards branch control information, the method comprising the steps of:

in a FILL state

fetching instruction packets from a primary memory unit,

writing a copy of said instruction packets into a DLB,

setting a valid bit to assert validity of said DLB,

entering an ACTIVE state when a backward branch is taken within a range of a loop,

entering an IDLE state when said backward branch completes a loop and a change of flow branch is taken out of range of said loop, and

entering an OVERFLOW state if the DLB fills up and said change of flow branch is not taken;

in the ACTIVE state

presenting instruction fetches to the DLB,

entering the IDLE state if said backward branch is not taken and a change of flow branch is taken out of range,

entering the FILL state if the buffer is not full, if said backward branch is within the loop,

entering an OVERFLOW state if the buffer is full,

entering the FILL state if the buffer is full, and

entering the OVERFLOW state if the end of the buffer is reached said and change of flow branch is not taken;

in the OVERFLOW state

reading instruction data packets from a primary memory unit of said computing device,

entering the IDLE state after performing a jump or a branch instruction out of range of said loop;

entering the ACTIVE state if said backward branch is taken,

entering the ACTIVE state if a backward branch within the range of said loop is taken and data is valid, and

entering the FILL state if a backward branch within the range of said loop is taken and data is invalid; and

in the IDLE state

accessing instruction data packets from the primary memory unit,

entering the FILL state if a new backward branch is detected, and

entering the ACTIVE state if a last loop captured in the dynamic loop buffer is repeated.